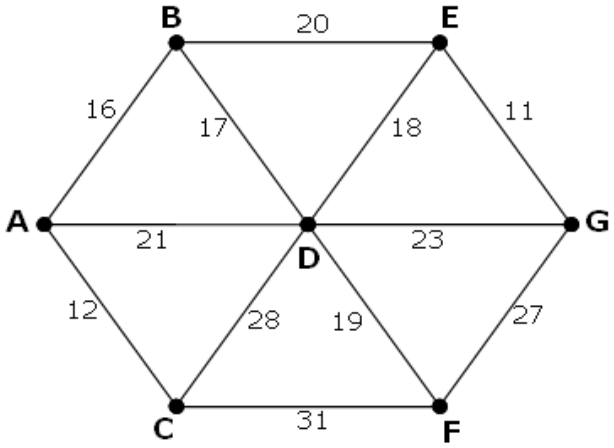
Algorytmy i złożoność obliczeniowa

Laboratorium 8: Grafy. Znajdowanie najkrótszej drogi w grafie ważonym.

1. Graf ważony

Graf ważony to graf, w którym każdej krawędzi przypisana jest pewna **liczba**- **waga** (może ona oznaczać na przykład odległość między wierzchołkami). Poniżej mamy przykład nieskierowanego grafu ważonego, który składa się z siedmiu wierzchołków i dwunastu krawędzi oraz jego reprezentację macierzową.



	A	B	C	D	E	F	G
A	-	16	12	21	-	-	-
B	16	-	-	17	20	-	-
C	12	-	-	28	-	31	-
D	21	17	28	-	18	19	23
E	-	20	-	18	-	-	11
F	-	-	31	19	-	-	27
G	-	-	-	23	11	27	-

Dla podanego grafu macierz *A* dla każdej pary wierzchołków u i v zawiera wagę krawędzi (u,v), a jeśli krawędź (u,v) nie istnieje, to przyjmujemy, że jej waga wynosi nieskończoność i wpisujemy * lub -.

2. Algorytm Forda-Bellmana

Algorytm ten służy do wyznaczania najmniejszej odległości od ustalonego wierzchołka ***s*** do wszystkich pozostałych w grafie skierowanym bez cykli o ujemnej długości (uwaga! dopuszczalne są ujemne wagi). Warunek nieujemności cyklu jest spowodowany faktem, że w grafie o ujemnych cyklach najmniejsza odległość między niektórymi wierzchołkami jest nieokreślona, ponieważ zależy od liczby przejść w cyklu.

Algorytm Forda-Bellmana w każdym kroku oblicza górne oszacowanie $D(v_i)$ odległości od wierzchołka ***s*** do wszystkich pozostałych wierzchołków v_i. W pierwszym kroku przyjmujemy $D(v_i)=A(s,v_i)$. Gdy stwierdzimy, że **D(v)>D(u)+A(u,v)**, to każdorazowo polepszamy aktualne oszacowanie i podstawiamy **D(v):=D(u)+A(u,v).** Algorytm kończy się, gdy żadnego oszacowania nie można już poprawić, macierz $D(v_i)$ zawiera najkrótsze odległości od wierzchołka ***s*** do wszystkich pozostałych.

Pseudokod algorytmu Forda-Bellmana (n- liczba wierzchołków)

```
for v:=[1..n] do D(v):=A(s,v);      // w pierwszym kroku przyjmujemy, że najkrótszą drogą z u do v jest krawędź (u,v)
for k:=1 to (n-2) do      //w (n-2) krokach można zbadać każdą drogę o (n-1) krawędziach i wybrać najkrótszą
      begin
        for v:=[1..n]-{s} do            //odległość od s do s wynosi 0 i już się na pewno nie zmieni
          for u:=[1..n]-{s} do
            D(v):=min{D(v),D(u)+A(u,v)};          //wybierz krótszą drogę
      end;
```

3. Algorytm Dijkstry

Algorytm ten służy do wyznaczania najmniejszej odległości od ustalonego wierzchołka ***s*** do wszystkich pozostałych w skierowanym **grafie**, w odróżnieniu jednak od algorytmu Forda-Bellmana, graf wejściowy nie może zawierać krawędzi o ujemnych wagach.

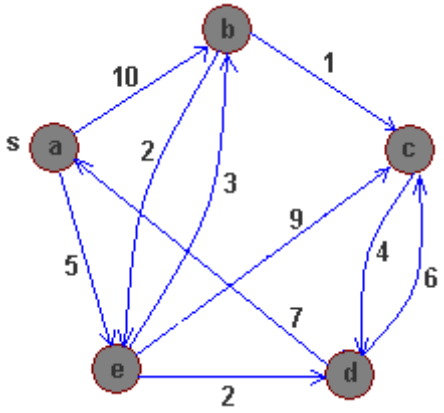
W algorytmie tym pamiętany jest zbiór **Q** wierzchołków, dla których nie obliczono jeszcze najkrótszych ścieżek, oraz wektor **D[i]** odległości od wierzchołka ***s*** do ***i***. Początkowo zbiór **Q** zawiera wszystkie wierzchołki a wektor **D** jest pierwszym wierszem macierzy wag krawędzi **A**.

Algorytm przebiega następująco

1. Dopóki zbiór Q nie jest pusty wykonuj:
2. Pobierz ze zbioru Q wierzchołek *v* o najmniejszej wartości D[v] i usuń go ze zbioru.
3. Dla każdego następnika *i* wierzchołka *v* dokonaj relaksacji ścieżki, tzn. sprawdź, czy D[i]>D[v]+A[v,i], tzn. czy aktualne oszacowanie odległości do wierzchołka *i* jest większe od oszacowania odległości do wierzchołka *v* plus waga krawędzi (v,i). Jeżeli tak jest, to zaktualizuj oszacowanie D[i] przypisując mu prawą stronę nierówności (czyli mniejszą wartość).

Jak widać z powyższego pseudokodu, algorytm wybiera z kolejki Q "najlżejszy" wierzchołek, tzn. jest oparty o strategię **zachłanną**. Algorytm Dijkstry jest algorytmem dokładnym.

Przykład działania algorytmu Dijkstry (nieskończoność oznacza brak krawędzi):



	a	b	c	d	e
a	0	10	∞	∞	5
b	∞	0	1	∞	2
c	∞	∞	0	4	∞
d	7	∞	6	0	∞
e	∞	3	9	2	0

Obliczenia przebiegają następująco: (Dla s=a)
(gwiazdka oznacza symbol nieskończoności, najlżejszy wierzchołek jest podkreślony, wierzchołki, dla których wyznaczono już najkrótsze ścieżki są pogrubione)

Q	D(a)	D(b)	D(c)	D(d)	D(e)
{b,c,d,e}	**0**	10	*	*	5
{b,c,d}	**0**	8	14	7	**5**
{b,c}	**0**	8	13	**7**	**5**
{c}	**0**	**8**	9	**7**	**5**
{}	**0**	**8**	**9**	**7**	**5**

4. Algorytm Floyda-Warshalla

Algorytm ten podobnie jak algorytm Dijkstry, wykorzystuje macierz kosztów, ale jej elementami mogą być również liczby ujemne. Algorytm działa w oparciu o strategię programowania dynamicznego, wyznaczając długości najkrótszych ścieżek w tablicy dwuwymiarowej n×n, powiedzmy o nazwie **D**.
Początkowo wartości elementów **D[u, v]** są równe wagom **A[u, v],** a w kolejnych **n** iteracjach są korygowane:

Pseudokod algorytmu Floyda-Warshalla (n- liczba wierzchołków)

```
for u:=1 to n do
  for v:=1 to n do
    D[u,v] := A[u,v];
  D[u,u] := 0 ;
for k:=1 to n do
  for u:=1 to n do
    for v:=1 to n do
      D[u,v] := min(D[u,v], D[u,k]+D[k,v])
```

Jeżeli w **k**-tej iteracji najkrótsza znana ścieżka wiodąca od wierzchołka **u** do **v** jest dłuższa niż ścieżka od **u** do **v** przechodząca poprzez pośredni wierzchołek **k**, to jako nowa najkrótsza ścieżka od **u** do **v** jest wybierana ta, która wiedzie przez wierzchołek **k**. Mówiąc dokładniej, po pierwszej iteracji wartością **D[u, v]** jest długość najkrótszej ścieżki wiodącej bezpośrednio od wierzchołka **u** do **v** lub pośrednio przez wierzchołek **1**, po drugiej iteracji – długością najkrótszej ścieżki od **u** do **v** przechodzącej co najwyżej przez wierzchołki pośrednie ze zbioru {1, 2} itd. Ogólnie, po **k**-tej iteracji **D[u, v]** jest długością najkrótszej ścieżki, która wiedzie od **u** do **v** i nie zawiera innych wierzchołków pośrednich niż {1, 2, ..., k}.
Po zakończeniu wszystkich iteracji wartościami elementów **D[u, v]** są długości najkrótszych ścieżek od **u** do **v** przebiegających przez wierzchołki zbioru **V**.

5. Interfejs grafu ważonego

Należy pobrać plik IWGraph.java bezpośrednio ze strony WWW.

6. Zadania

1. W oparciu o podany interfejs zaimplementować metody: addVertex, addEdge, Wcheck, writeGraph dla grafu ważonego. Napisać krótki program wykorzystujący ww. metody, który zaprezentuje ich działanie. **[3p]**
2. Korzystając z algorytmu Dijkstry zaimplementować metodę dijkstra znajdowania najkrótszej drogi w grafie ważonym. Efektem działania tej metody powinna być tabela pokazująca poszczególne etapy działania algorytmu (patrz powyżej na przykład działania algorytmu). **[3p]**
3. Korzystając z algorytmu Bellmana-Forda zaimplementować metodę ford znajdowania najkrótszej drogi w grafie ważonym. **[2p]**
4. Zaimplementować metodę wypisującą silnie spójne składowe. **[2p]**

Praca domowa

Zaimplementować metodę znajdowania najkrótszych dróg w grafie ważonym za pomocą algorytmu Floyda-Warshalla.

Na następnych zajęciach: Drzewa, drzewa BST